ISSN 2395-1621

A Survey: Dynamic Allocation of Slots for Map Reduce Workload.

[#1]Dhananjay A. Ahire, [#2]Narayan K. Dongare, [#3]Pooja P. Wankhede,
[#4]Chiranjivi D. Mane, [#5]Prof. Devendra Gadekar

[1]ahire.dhunanjay12@gmail.com
[2]narayandongare7784@gmail.com
[3]poojaw2903@gmail.com

[#1234]Department of Computer Engineering,
[#5]Prof. Department of Computer Engineering,

JSPM's
Imperial College Of Engineering & Research, Wagholi
Pune, India.

ABSTRACT

MapReduce is a current computing standard for important large data processing in one or more clusters. A MapReduce system having a set of jobs and each job having a multiple map and multiple reduce tasks because of these 1) in map slot, map task are get run and in reduce slot, reduce task are get run, and 2) reduce tasks are executed after map tasks, execution orders of job is different and map/ reduce slot configurations for different system have a different execution performance and utilization. But those existing techniques in the process are not providing the better performance because of the un-improved resource allocation. To overcome this drawback, and to improve a performance of resource allocation with the proposed technique called DynamicMR. this concept includes a two phases of algorithm is MK_TCT (makespan and the total completion time) for an offline workload and second phase of algorithm for optimized resource allocation . In these system to consider a three major steps to improve a system performance and resource utilization for basic MapReduce technique, 1) DHSA- Dynamic Hadoop Slot Allocation (i.e it used to overcome the slot allocation restraint), 2)SEPB- Speculative Execution Performance Balancing (i.e it used to steadiness the performance in cluster of jobs), 3)SP-Slot Pre-scheduling (i.e it used to advance the data vicinity). Micro-level optimization is used as a optimization approach, it is used to improve idle slot utilization efficiency after the Macro-level optimization. Particularly, we identify that three main affecting factors: Speculative tasks, dynamic slot allocation, and pre-scheduling.

Keywords: **MapReduce, Dynamic Hadoop Slot Allocation [DHSA], Speculative Execution Performance Balancing [SEPB], Slot Pre-scheduling, MK_TCT, DynamicMR , Optimized Resource Allocation.**

ARTICLE INFO

Article History

Received: 1st December 2016

Received in revised form :

2nd December 2016

Accepted: 5th December 2016

Published online :

6th December 2016

I. INTRODUCTION

Over the past five years, the authors and many others at Google have executed hundreds of special-purpose multiplications that process large amounts of raw data, such as crawled papers, web request logs, etc., to compute various kinds of derived data. However, the input data is usually large and the deductions have to be distributed across hundreds or thousands of machines in order to surface in areas on able quantity of time. The subjects of how to parallelize the calculation, allocate the data, and handle failures conspire to obscure the original simple calculation with large quantities of multifaceted code to deal with these issues. Firstly, the compute resources (e.g., CPU cores) are abstracted into map and reduce slots, which are basic compute units and statically configured by administrator in advance [1]. A MapReduce job execution has two unique features: 1) the slot allocation constraint assumption that map slots can only be allocated to map tasks and reduce slots can only be allocated to reduce tasks, and 2) the general execution constraint that map tasks are executed before reduce tasks. Due to these

features, we have two observations: [1]. there are significantly different performance and system utilization for a MapReduce workload under different slot configurations, and [2]. even under the optimal map/reduce slot configuration, there can be many idle reduce (or map) slots while map (or reduce) slots are not enough during the computation, which adversely affects the system utilization and performance. Speculative execution is an important technique that can overcome the problem of slow-running task's influence on a single job's execution time by running a backup task on another machine. However, it comes at the cost of cluster efficiency for the whole jobs due to its resource competition with other running tasks. We propose a dynamic scheduling and slot allocation policy for speculative task. It balances the tradeoff between a single job's execution time and a batch of jobs' execution time by determining dynamically when it is time to schedule and allocate slots for speculative tasks.

Delay scheduling has been shown to be an effective approach for the data locality improvement in MapReduce [3]. It achieves better data locality by delaying slot assignments in jobs where there are no currently local tasks available. However, it is at the cost of fairness. In contrast, we propose a scheduler named Pre-Scheduler that can improve the data locality while not hurt the fairness. It considers the case of a node where there are currently local map tasks of jobs and idle slots available, but no allowable idle map slots (e.g., due to the load balancing constrain) to be allocated. It pre-allocates idle slots of the node to jobs to maximize the data locality and guarantee fairness. Since Delay scheduling and Pre-scheduling works in different scenarios, our DLMS incorporates both approaches, making them work cooperatively to maximize the data locality.

We propose DynamicMR, a dynamic scheduling framework for MapReduce, in order to improve the utilization and performance of a shared Hadoop cluster under a fair scheduling between users. Figure 1 gives an overview of DynamicMR. It consists of three levels of scheduling components, i.e., Dynamic Hadoop Fair Scheduler (DHFS),[9] Dynamic Speculative Task Scheduler (DSTS) , and Data Locality Maximization Scheduler (DLMS) .Each scheduler considers the performance improvement from different aspects. DHFS attempts to maximize slot Utilization as possible while guarantee the fairness, when there are pending tasks (e.g., map tasks or reduce tasks).DSTS identifies the slot resource in-efficiency problem for a Hadoop cluster, brought by speculative tasks.

II. EXISTING SYSTEM STUDY

Existing work focus on the full slot utilization for map while reduce slots are blank and vice versa. Hence they are cruelly underutilized. They also consider the speculative execution efficiency to be high in the single job only. They did not consider the cluster efficiency in the above consideration. The data locality maximization is important for the slot utilization efficiency and MapReduce workloads performance improvement. But there is conflict between the fairness and data locality optimization.

The spectrum of deployed wireless cellular communication systems is found to be under-utilized, even though licensed spectrum is at a premium. In this paper, we design a system with an ad hoc overlay network, which we denote as the secondary system (SEC), to efficiently utilize the bandwidth left unused in a cellular system, which we denote as the primary system (PRI). The basic design principle is that the SEC operates in a non-intrusive manner and does not interact with the PRI. We develop the AS-MAC, [7] an Ad hoc SEC Medium Access Control protocol to enable the interoperation of the PRI-SEC system. We address a number of technical challenges pertinent to this networking environment, and investigate a number of AS-MAC variants. Our performance evaluation results indicate that AS-MAC can transparently utilize up to 80% bandwidth left unused by the PRI.

The input files into multiple map tasks, and then schedules both map tasks and reduce tasks to worker nodes in a cluster to achieve parallel processing. A machine takes an unusually long time to complete a task. Delay the job execution time and degrade the cluster throughput significantly. The Existing system is handled via speculative execution slow task is backed up on an alternative machine.

For the processing of medical data we use the above three techniques. Using the above techniques Here Whenever there is an idle slot available, Dynamic MR will first attempt to improve the slot utilization with DHSA. Decisions are taken dynamically for allocation based on constraints. If the allocation is true, Dynamic MR will further optimize the performance by improving the efficiency of slot utilization with SEPB. Dynamic MR will be able to further improve the slot utilization efficiency from the data locality optimization aspect with Slot Prescheduling.

To determine an optimized path to perform the sequence of MapReduce jobs and uses Hadoop MapReduce clusters deployed in each of the considered data centers to execute the determined optimized path accordingly. Job Managers perform the jobs accordingly using the Hadoop MapReduce clusters deployed in corresponding data centers. A Distributed Map Reduce (D-MR) framework is proposed for efficient job execution with optimized resource consumption.

Disadvantages:

1. Use average progress rate to identify slow tasks while in reality the progress rate can be unstable and misleading.

2. We cannot appropriately handle the situation when there exists data skew among the tasks.

3. Do not consider whether backup tasks can finish earlier when choosing backup worker nodes.

4. Job execution time and cluster throughput are high. The average progress is slow task for execute the large scale data processing.

III. BASIC IDEA

A. Map-reduced

Map Reduce is a processing technique and a program model for distributed computing based on java it contains two important tasks, namely Map and Reduce. The major advantages of MapReduce is that it is easy to scale data processing over multiple computing nodes [9].

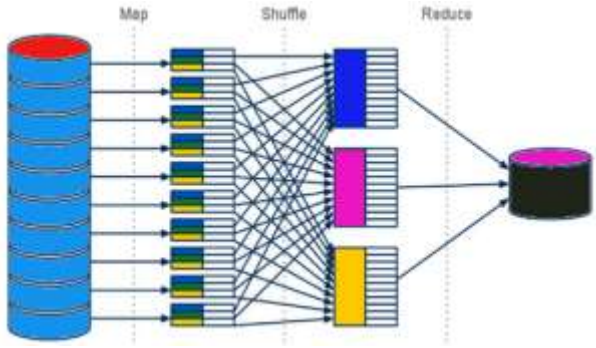

Fig.1:MapReduce Working

B. Hadoop Distributed File System(HDFS)

Hadoop File System was developed using distributed file system design. It is run on commodity hardware. Unlike other distributed systems, HDFS is highly fault tolerant and designed using low-cost hardware. [8]
HDFS holds very large amount of data and provides easier access. To store such huge data, the files are stored across multiple machines. These files are stored in redundant fashion to rescue the system from possible data losses in case of failure. HDFS also makes applications available to parallel processing.

Features of HDFS

1. It is suitable for the distributed storage and processing.
2. Hadoop provides a command interface to HDFS.
3. Streaming access to file system data.
4. HDFS provides file permissions and authentication.

Given below is the architecture of a Hadoop File System.
HDFS

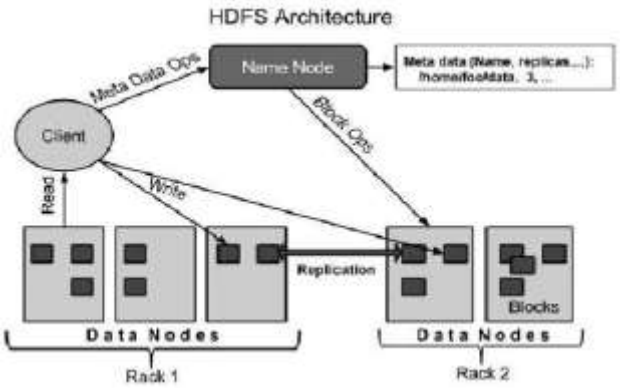

Fig.2: HDFS Architecture

Follows the master-slave architecture and it has the following elements:

1. Name node:
The name node is the commodity hardware that contains the GNU/Linux operating system and the name node software. It is a software that can be run on commodity hardware. The system having the name node acts as the master server and it does the following tasks:

2. Data node:
The data node is a commodity hardware having the GNU/Linux operating system and data node software. For every node (Commodity hardware/System) in a cluster, there will be a data node. These nodes manage the data storage of their system.
Data nodes perform read-write operations on the file systems, as per client request. They also perform operations such as block creation, deletion, and replication according to the instructions of the name node.

3. Block:
Generally the user data is stored in the files of HDFS. The file in a file system will be divided into one or more segments and/or stored in individual data nodes. These file segments are called as blocks. In other words, the minimum amount of data that HDFS can read or write is called a Block. The default block size is 64MB, but it can be increased as per the need to change in HDFS configuration.

C. Data preprocessing

Imputation is the process of replacing missing data with substituted values. Single imputation-A once-common method of imputation was hot-deck imputation where a missing value was imputed from a randomly selected similar record. In cases with imputation, there is guaranteed to be no relationship between the imputed variable and any other measured variables. Thus, mean imputation has some attractive properties for univariate analysis but becomes problematic for multivariate analysis.

D. Dynamic Hadoop Slot Allocation(DHSA)

In contrast to YARN which proposes a new resource model of container that both map and reduce tasks can run on, DHSA keeps the slot-based resource model. The idea for DHSA is to break the assumption of slot allocation constraint to allow that: 1) Slots are generic and can be used by either map or reduce tasks, although there is a pre-configuration for the number of map and reduce slots. In other words, when there are insufficient map slots, the map tasks will use up all the map slots and then borrow unused reduce slots Similarly, reduce tasks can use unallocated map slots if the number of reduce tasks is greater than the number of reduce slots. 2)Map tasks will prefer to use map slots and likewise reduce tasks prefer to use reduce slots. The benefit is that, the pre-configuration of map and reduce slots per slave node can still work to control the ratio of running map and reduce tasks during runtime, better than YARN which has no control mechanism for the ratio of running map and

reduce tasks. The reason is that, without control, it easily occurs that there are too many reduce tasks running for data shuffling, causing the network to be a bottleneck seriously

E. Speculative Execution Performance Balancing (SEPB)

When a node has an idle map slot, we should choose pending map tasks first before looking for speculative maptasks for a batch of jobs. Hadoop Slot is executed for determining path for performing the MapReduce job. After this the Speculative based process starts to execute the determined optimized Multi-execution path. Executing individual MapReduce jobs in each datacenter on corresponding inputs and then aggregating results is defined as MULTI execution path. This path used to execute the jobs effectively.

F. Slot pre-scheduling(SP)

In this module we are going to perform two processes. Slot allocation Slot pre-scheduling process. In this slot allocation process we are going allocate the slot based on dynamic Hadoop slot allocation optimization mechanism. In the slot pre-scheduling process we are going to improve the data locality. Slot Pre-Scheduling technique that can improve the data locality while having no negative impact on the fairness of Map-Reduce jobs. Some idle slots which cannot be allocated due to the load balancing constrain during runtime, we can pre-allocate those slots of the node to jobs to maximize the data locality.

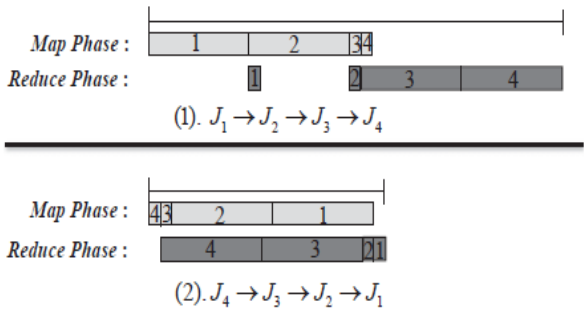

Fig.3: Slot Scheduling for map and reduce task

G. **Performance Evaluation:**

Speculative execution is a common approach for dealing with the straggler problem by simply backing up those slow running tasks on alternative machines. Multiple speculative execution strategies have been proposed, but there is a pitfall: incoming jobs are allocated to nodes present in server and fail to schedule process type allocate to node for processing. Performance is evaluated by means of selective the optimized resources and results taken in terms of execution time, processing memory.

IV. USEFUL ALGORITHMS:

A. Make Span Optimization
B. MK_TCT
C. MK_JR
D. FIFO
E. Job Ordering Optimization

V. CONCLUSION

The aim of the proposed system is to improve the performance of Map Reduce workloads. It considered three techniques: Dynamic Hadoop Slot Allocation Speculative Execution Performance Balancing, and Slot Pre-scheduling . Dynamic Hadoop Slot Allocation use allocation of map to maximize the slot utilization and it reduces the task dynamically

REFERENCES

[1] J. Dean and S. Ghemawat. MapReduce: Simpli_ed Data Processing on Large Clusters, In Proceedings of the 6th Symposiumon Operating SystemsDesign and Implementation (OSDI), 2004.
[2] Hadoop. http://hadoop.apache.org.
[3] B. Moseley, A. Dasgupta, R. Kumar, T. Sarl, On scheduling in map-reduce and ow-shops. SPAA, pp. 289-298, 2011.
[4] A. Verma, L. Cherkasova, R. Campbell. Two Sides of a Coin: Optimizing the Schedule of MapReduce Jobs to Minimize Their Makespan and Improve Cluster Performance. MASCOTS 2012.
[5] J. Dittrich, J.-A. Quiane-Ruiz, A. Jindal, Y. Kargin, V. Setty, and J. Schad. Hadoop++: Making a Yellow Elephant Run Like a Cheetah, PVLDB, 3(1), 2010.
[6] D.W. Jiang, B.C. Ooi, L. Shi, and S. Wu.The Performance of MapReduce: An Indepth Study, PVLDB, 3:472-483, 2010.
[7] T. Nykiel, M. Potamias, C. Mishra, G. Kollios, and N. Koudas. MRShare: Sharing Across Multiple Queries in MapReduce Proc. of the 36th VLDB (PVLDB), Singapore, September 2010.
[8]HowManyMapsAndReduces.http://wiki.apache.org/hadoop/HowManyMaps An-dReduces.
[9] M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy,S. Schenker,I. Stoica, Job Scheduling for Multi-user Mapreduce Clusters. Technical Report EECS-2009-55, UC Berkeley Technical Report (2009).
[10] http://hadoop.apache.org/common/docs/r0. 20.1/capacity scheduler.html, Capacity Scheduler Guide. 2010.
[11] J.N.D. Gupta, Two stage hybrid owshop scheduling problem, Journal of Operational Research Society 39 (4) (1988) 359C364.
[12] W. Cirne and F. Berman, When the herd is smart: aggregate behavior in the selection of job request, IEEE Transactions on Parallel and Distributed Systems, vol. 14, pp.181-192, 2003.
[13] S.M. Johnson. Optimal two- and three-stage production schedules with setup times included. Naval Res Logist Q. vol. 1, pp. 61-68, 1954.
[14] P. Agrawal, D. Kifer, and C. Olston. Scheduling Shared Scans of Large Data Files. In VLDB, 2008.

[15] http://en.wikipedia.org/wiki/Lognormal distribution.
[16] Amazon EC2. http://aws.amazon.com/ec2.

[17] M. Zaharia, D. Borthakur, J. Sarma, K. Elmeleegy,S. Schenker,I. Stoica, Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In Proceedings of EuroSys
[18] http://hadoop.apache..org to understand a basic things for projects.
[19] A. Verma, L. Cherkasova, and R. H. Campbell. Play It Again, SimMR! In Proc.of Intl. IEEE Cluster 2011 (Cluster2011). Austin, TX, USA, Sept, 2011.
[20]http://riot.ieor.berkeley.edu/Applications/Schedu ling/algorithms.html, The Scheduling Problem.
[21] K. Howard, S. Siddharth and V. Sergei. A model of computation for MapReduce, Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 938-948, 2010.
[22] P.F. Dutot, G. Mounie, and D. Trystram. Scheduling parallel tasks approximation algorithms. In J.T. Leung (ed.), Handbook of Scheduling: Algorithms, Models, and Performance Analysis. Chapman Hall, CRC Press,2004.
[23] P. Sanders, J. Speck. E_cient Parallel Scheduling of Malleable Tasks. IPDPS,pp. 1156-1166, 2011.
[24] T. Condie, N. Conway, P. Alvaro, J.M. Hellerstein. MapReduce online. In Proceedings of the 7th USENIX conference on Networked systems design and imple-mentation, pp. 21C21, 2010.
[25] P. Dutot, L. Eyraud, G. Mounie, D. Trystram. Bi-criteria Algorithm for Schedul-ing Jobs on Cluster Platforms, SPAA, pp. 125-132, 2004.
[26] J.Y.T. Leung, Handbook of Scheduling: Algorithms, Models, and PerformanceAnalysis, Chapman and Hall/CRC, 2004, 25-5-25-18.
[27] G. J. Kyparisis, C. Koulamas. A note on makespan minimization in twostage exible ow shops with uniform machines. European Journal of Operational Re-search, Vol 175, pp. 1321-1327, 2006.
[28] S.R. Hejazi, S. Sagha_an. Flowshop-scheduling problems with makespan crite-rion: a review, International Journal of Production Research, vol. 43, pp. 2895-2929,2005.
[29] A. Ganesh, K. Srikanth, G. Albert, S. Ion, Lu. Yi, S. Bikas and H. Edward.
Reining in the outliers in map-reduce clusters using Mantri, OSDI10, pp.1-16, 2010.

